MLCompare: A Facilitating Framework for Machine Learning Research

Victoria Stodden vcs@stodden.net
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign

DataScience@HEP
FNAL, Batavia, IL
May 9, 2017

NCSA

Agenda

1. Computational Reproducibility and the Scholarly Record
2. A Case Study: Resolving Discrepancies in Genome-based Disease Classification
3. Introducing MLCompare: A Framework for Extensible Machine Learning Research

In the Future: A Reproducible Scholarly Record

We claim:

- Digital scholarly objects needed to reproduce and verify findings will be available with the published claim (i.e. source data, tuning parameters, source code and algorithm implementation, workflows, ...)
- This will imply novel interactions with the scholarly record that advance scientific discovery.

In the Future: Querying a Reproducible Scholarly Record

- List all of the image denoising algorithms ever used to remove white noise from the famous “Barbara” image, with citations;
- List all of the classifiers applied to the famous acute lymphoblastic leukemia dataset, along with their misclassification rates;
- Create a unified dataset containing all published whole-genome sequences identified with mutation in the gene BRCA1;
- Randomly reassign treatment and control labels to cases in published clinical trial X and calculate effect size. Repeat many times and create a histogram of effect sizes. Do this for all clinical trials published in 2003 and list the trial name and histogram side by side.

Courtesy Donoho and Gavish, 2012

The Acute Lymphoblastic Leukemia Dataset

Introduced in Golub et al. “Molecular classification of cancer: class discovery and class prediction by gene expression monitoring” (1999): “*cancer classification based on gene expression monitoring by DNA microarrays is described and applied to human acute leukemias [to] discover the distinction between acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL)*”

In joint work with Xiaomian Wu and April Tang, we tried the scholarly record query.


The Acute Lymphoblastic Leukemia Dataset Query

We wanted:

- A list of all classifiers applied to the Golub dataset;
- A comparison of their misclassification rates.

A literature search produced 30 articles, but they did not give comparable misclassification rates.

Our next step was to create the table of misclassification rates. We identified 5 articles for which this seemed possible.

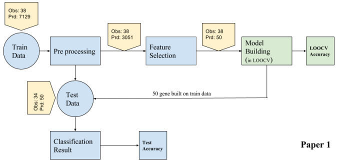
Our (Naive) Expectation

We obtained the original Golub data. We hoped to apply the various machine learning algorithms from the literature, in the 5 cases we identified.

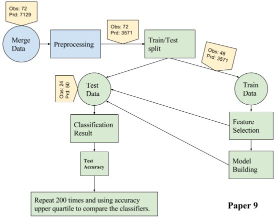
We found that the articles implemented (at least) three steps, each varying from one article to the next:

1. data preprocessing,
2. feature selection,
3. application of machine learning algorithm.

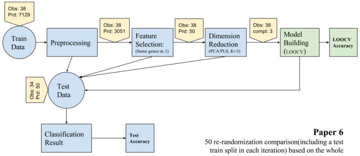
Computational Steps in the 5 Chosen Articles



Paper 1

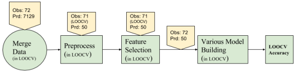


Paper 9



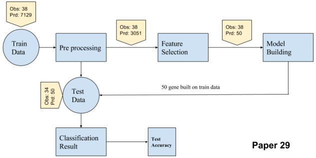
Paper 6

50 re-randomization comparison(including a test train split in each iteration) based on the whole process also conducted



Paper 3

To compare with Golub. Paper, we use the train data to train and test data to test in our reproduce work. Hence, our gene selection dataset for paper3 is based on train data and no LOOCV performed.



Paper 29

Learning Algorithms Applied (typically $47 ALL, 25 AML$)

Paper	Data Size	Algorithm(s) Applied
1	72×6817	Golub Classifier: informative genes+weighted vote
2	72×6817	Golub Classifier: informative genes+weighted vote
3	72×7129	Nearest Neighbor; SVM(linear kernel, quadratic kernel); Boosting (100, 1000, 10000 iterations)
4	72×7129	SVM(top 25, 250, 500, 1000 features)
5	72×7070	MVR(median vote relevance); NBGR(naive bayes global relevance); MAR(Golub relevance)+SVM
6	72×6817	Logistic and Quadratic discriminant analysis
7	72×7129	SVM
9	72×6817	Linear and Quadratic discriminant analysis; Classification trees; NN
10	72×7129	Decision Trees; AdaBoost
11	72×7129	MAVE-LD, DLDA, DQDA, MAVE-NPLD
12	72×7129	SIMCA classification
...	...	...

Classification Efficiencies: Algorithm × Feature Selection

	1	3	6PCA	6PLS	9	29
Paper1 Classifier	0.912	0.941	0.971	0.971	0.958	0.706
Paper3 NN	0.971	0.941	0.912	0.941	1	0.912
Paper3 SVM Linear	0.971	0.971	0.941	0.971	1	0.765
Paper3 SVM Quadratic	0.971	0.882	0.971	0.971	1	0.912
Paper3 Adaboost	0.912	0.912	0.971	0.971	0.958	0.941
Paper6 PCA logit	0.971	0.971	0.971		1	0.853
Paper6 PCA QDA	0.941	0.912	0.941		1	0.853
Paper6 PLS logit	0.971	0.882		0.971	1	0.853
Paper6 PLS QDA	0.971	0.882		0.971	1	0.853
Paper9 NN	0.971	0.912	0.853	0.971	0.958	0.971
Paper9 Decision Tree	0.912	0.912	0.971	0.971	0.917	0.735
Paper9 Bagging	0.971	0.912	0.971	0.971	0.958	0.735
Paper9 Bagging (CPD)	0.941	0.912	0.971	0.971	0.917	0.794
Paper9 LDA	0.912	0.912	0.971	0.971	0.958	0.794
Paper9 Diagonal LDA	0.941	0.912	0.971	0.971	0.958	0.765
Paper9 Diagonal QDA	0.912	0.912	0.971	0.971	0.958	0.735
Paper29 Bayesian Network	0.735	0.882	0.971	0.971	1	0.647

Conclusion

- Hard to synthesize (200+ student hours)
- Many points of variability: starting dataset; preprocessing steps; feature selection methods; algorithm choice; tuning of algorithm...
- Details not well-captured in the traditional article, making comparisons difficult or impossible.

Would be easier if:

- there was prior agreement on the dataset,
- prior agreement on hold-out data for testing,
- full disclosure of feature selection steps,
- full disclosure of algorithm application and parameter tuning.

The “CompareML Framework”

Adapt the Common Task Framework from Natural Language Processing: “CompareML Framework”

- Agreement on datasets prior to analysis, conferences around those datasets,
- Hold-out data held by a neutral third party (NIST), not seen by researchers,
- Researchers distinguish and specify feature selection and preprocessing vs learning algorithm application,
- Send code to the third party who returns your misclassification rate on the test data.

Side effect: training data and code/algorithm shared.

The effect of infrastructure choices on the variability of a scientific result

Matthew Krafczyk krafczyk.matthew@gmail.com
National Center for Supercomputing Applications

DataScience@HEP
FNAL, Batavia, IL
May 9, 2017

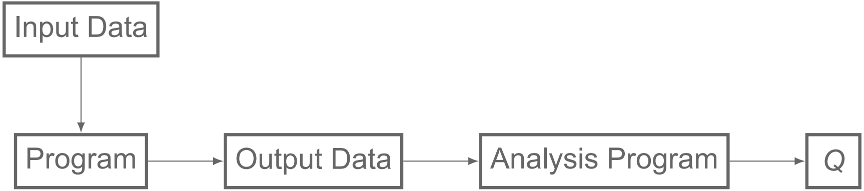
NCSA

Outline

- Some Definitions
- Preliminary Work
- Our Questions

Some Definitions

- Interested in programs which:
  - Produce output deterministically in single threaded mode.
  - Produce output non-deterministically in multi threaded mode.
  - Produce output data from which some scientific result can be extracted.



Problem Definition

- Interested in programs which:
  - Produce output deterministically in single threaded mode.
  - Produce output non-deterministically in multi threaded mode.
  - Produce output data from which some scientific result can be extracted.

$$\boxed{D_i} \longrightarrow \boxed{P(\cdot)} \longrightarrow \boxed{Q}$$

$$P(D_i) = Q \tag{1}$$

Some Definitions: Intrinsic Uncertainty

- $P(\cdot)$ is run multiple times with identical input data/initial conditions

$$\begin{tikzcd} \boxed{P(D_i)} \arrow[d, "1"] & \boxed{P(D_i)} \arrow[d, "2"] & \cdots & \boxed{P(D_i)} \arrow[d, "n"] \\ Q_1 & Q_2 & \cdots & Q_n \end{tikzcd}$$

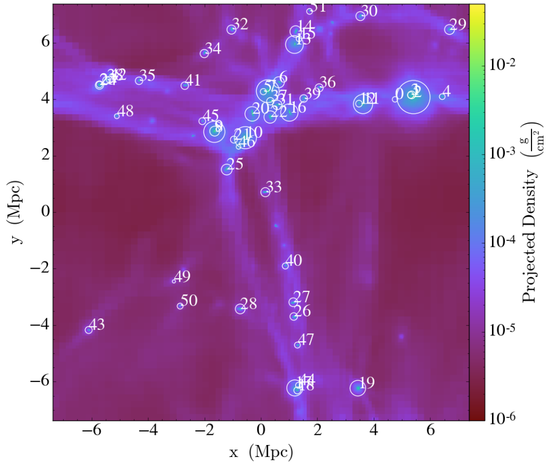
$$[P(D_i)]_j = Q_j \rightarrow \{P(D_i)\} = f(Q) \approx \overline{Q} \pm \Delta Q \qquad (2)$$

- We call ΔQ the **Intrinsic Uncertainty** of Q for this ensemble of computations.

Some Definitions

- The character of $f(Q)$ and in turn, $\overline{Q}$ and ΔQ may depend on many factors such as:
  - Compiler Vendor
  - Compiler Version
  - Compiler Optimization Settings
  - Number of Threads
  - Program Settings such as grid resolution, or static/adaptive
  - Network Traffic (If using mpi across multiple nodes)
  - Underlying system health
- Some of these factors the user has control over, and others the user does not.

- We wish to measure **Intrinsic Uncertainty** as a function of
  1. Compiler Choice
  2. Program Settings
  3. Execution Environment

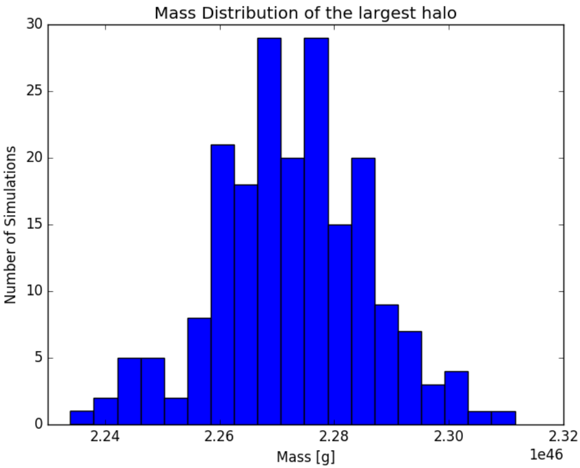

y (Mpc)
x (Mpc)
Projected Density $\left(\frac{g}{cm^2}\right)$
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}

Preliminary Work: Enzo: Variation

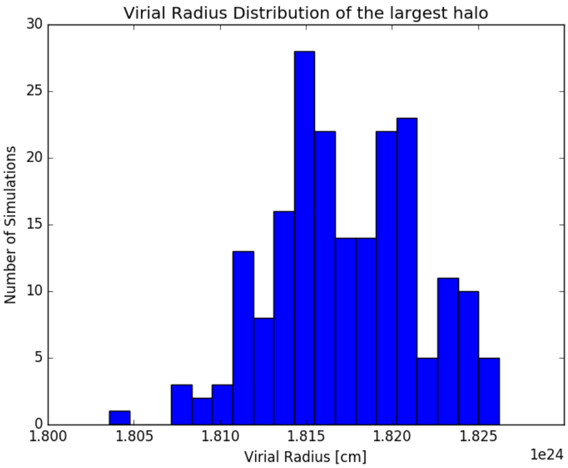
- Given identical initial conditions on Blue Waters, 200 runs of multithreaded Enzo produces varying results!

Show GIF!!

200 Simulations: Enzo: Variation



200 Simulations: Enzo: Variation



Preliminary Work

- Compilers
  - GCC (6.2.0)
  - Intel (16.0.3)
  - PGI (16.9.0)
- Compiler Optimization
  - -O0: No Optimization
  - -O1: Basic Optimization
  - -O2: High level of Optimization
  - -O3: Aggressive Optimization

Preliminary Work

Comp. Vendor	Comp. Version	Comp. Opt.	Time [hours]	Mass [$10^{43}g$] Mean	Mass [$10^{43}g$] Std. Dev.	Number Matched
gcc	6.2.0	-O0	$16.5 - 18.5$	2273	13(4)	200
		-O1	$6 - 8$	2268	14(4)	200
		-O2	$6.9 - 7.7$	2273	13(4)	200
		-O3	$6.8 - 7.8$	2273	14(4)	199
intel	16.0.3	-O0	$27.5 - 34$	2269	14(5)	200
		-O1	$5.9 - 6.6$	2270	13(4)	185
		-O2	$4.9 - 5.6$	2270	14(4)	200
		-O3	$5.0 - 5.7$	2270	14(4)	199
pgi	16.9.0	-O0	$10.6 - 12.1$	2270	14(4)	199
		-O1	$9.5 - 10.8$	2271	13(4)	200
		-O2	$7.5 - 8.6$	2271	13(4)	200
		-O3	$7.5 - 8.0$	2271	14(4)	198

Preliminary Work: Conclusions

- Intrinsic uncertainty does not appear to increase with increasing optimization in this application.
- Total intrinsic uncertainty in largest halo mass is small (0.5% effect)
- Matching algorithm had trouble with halo 2270 with some simulations! (Not a small effect!!)
- Next Steps: Role of MPI implementation, number of threads, and network communication

Our Questions

- Examples of simple (but multi-threadable) deterministic codes?
- Other sources of error? (at the infrastructure level)
- Do you know anybody else working on this?
- What is the relationship of this work to Uncertainty Quantification? (UQ)

Backup Slides

Future Work

- Can system faults adversly affect the output of scientific simulation without obvious signs?
  - Teaming up with the DEPEND group from UIUC to measure this
- Does intrinsic uncertainty change depending on the property measured?
- Measure intrinsic uncertainty with software other than Enzo

Blue Waters Info

- 22,640 Cray XE6 nodes (2 AMD 6276 “Interlagos” processors)
- 4,228 Cray XK7 nodes (1 AMD 6276 “Interlagos” processor with an NVIDIA GK110 (K20X) GPU
- Cray Gemini torus interconnect
- 300 PB Lustre filesystem

Work Done: Spack

- We use Spack to manage the build and run environment for Enzo
- Spack is new and some development was required for use on Blue Waters
- Development is still ongoing due to peculiarities with Cray’s compiler wrappers

Software Used: YT/rockstar

- We use YT to analyze the output of Enzo.
- Rockstar is the halo finder we used
- Both were compiled with Spack on Campus Cluster

Simulation details

- Each job run on blue waters was with a 32^3 root grid, with 7 levels of refinement, 10Mpc on a side and simulated from $z = 99$ (near beginning of universe) to $z = 0$ (present day) and 16 threads.
- Job completion times ranged from 30 hours to 4 hours. (Completable with one job)
- Halo analysis was performed on the Campus Cluster due to Cray compiler wrapper interference with analysis tools.
- This is a LOW RESOLUTION simulation.

Superhalo Finding

1. Halos are found using rockstar in each simulation.
2. Halos in different simulations are matched up using a ‘distance’ metric which balances mass, virial radius, and position.
3. If halos are matched consistently throughout all simulations, they are labelled as a ’superhalo’ and their properties can be studies across simulations.
4. Each simulation has about 50 labelled halos, but across all simulations there are about 35 super halos.